NPS-AM-12-C9P10R01-063

EXCERPT FROM THE

PROCEEDINGS

OF THE

NINTH ANNUAL ACQUISITION
RESEARCH SYMPOSIUM
WEDNESDAY SESSIONS
VOLUME I

An Innovative Approach to Lower the Risk of Software Intensive Development Programs

Jeff Dunlap
BAE Systems

Published April 30, 2012

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE **30 APR 2012**	2. REPORT TYPE	3. DATES COVERED **00-00-2012 to 00-00-2012**

4. TITLE AND SUBTITLE
An Innovative Approach to Lower the Risk of Software Intensive Development Programs

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
BAE Systems,Intelligence and Security,10920 Technology Pl ,San Diego,CA,92127

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR’S ACRONYM(S)

11. SPONSOR/MONITOR’S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Since 1973, nearly 80% of DoD ACAT I programs have experienced cost overruns, coupled with a four-out-of-five chance of not fielding capability to the warfighter on time. With the DoD acquisition reforms of the past two decades, the probability of program success (PoPS) rate is improving. To continue improving PoPS, program management tools and techniques need to develop and become institutionalized to monitor software-intensive-based capability control, and logic development efforts. A lack of sufficient tools to monitor software development costs and performance to the integrated baseline drives uncertainty and risk. The earned value management system (EVMS) approach lacks meaningful measures for software-intensive development programs and ?you can?t control what you can?t measure? (DeMarco, 1986, p. 58). Using commercially available (often freeware) tools, a robust set of automated managers can measure near real-time progress and identify trouble areas early in the software development process to allow meaningful correction to occur. This paper explores agile sprint development and continuous integration and test best practices, and the potential for an innovative approach of intertwining the two to reduce risk and increase PoPS.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**37**	

The research presented at the symposium was supported by the acquisition chair of the Graduate School of Business & Public Policy at the Naval Postgraduate School.

To request defense acquisition research or to become a research sponsor, please contact:

NPS Acquisition Research Program
Attn: James B. Greene, RADM, USN, (Ret.)
Acquisition Chair
Graduate School of Business and Public Policy
Naval Postgraduate School
Monterey, CA 93943-5103
Tel: (831) 656-2092
Fax: (831) 656-2253
E-mail: jbgreene@nps.edu

Copies of the Acquisition Research Program's sponsored research reports may be printed from our website (www.acquisitionresearch.net).

Preface & Acknowledgements

Welcome to our Ninth Annual Acquisition Research Symposium! This event is the highlight of the year for the Acquisition Research Program (ARP) here at the Naval Postgraduate School (NPS) because it showcases the findings of recently completed research projects—and that research activity has been prolific! Since the ARP's founding in 2003, over 800 original research reports have been added to the acquisition body of knowledge. We continue to add to that library, located online at www.acquisitionresearch.net, at a rate of roughly 140 reports per year. This activity has engaged researchers at over 60 universities and other institutions, greatly enhancing the diversity of thought brought to bear on the business activities of the DoD.

We generate this level of activity in three ways. First, we solicit research topics from academia and other institutions through an annual Broad Agency Announcement, sponsored by the USD(AT&L). Second, we issue an annual internal call for proposals to seek NPS faculty research supporting the interests of our program sponsors. Finally, we serve as a "broker" to market specific research topics identified by our sponsors to NPS graduate students. This three-pronged approach provides for a rich and broad diversity of scholarly rigor mixed with a good blend of practitioner experience in the field of acquisition. We are grateful to those of you who have contributed to our research program in the past and hope this symposium will spark even more participation.

We encourage you to be active participants at the symposium. Indeed, active participation has been the hallmark of previous symposia. We purposely limit attendance to 350 people to encourage just that. In addition, this forum is unique in its effort to bring scholars and practitioners together around acquisition research that is both relevant in application and rigorous in method. Seldom will you get the opportunity to interact with so many top DoD acquisition officials and acquisition researchers. We encourage dialogue both in the formal panel sessions and in the many opportunities we make available at meals, breaks, and the day-ending socials. Many of our researchers use these occasions to establish new teaming arrangements for future research work. In the words of one senior government official, "I would not miss this symposium for the world as it is the best forum I've found for catching up on acquisition issues and learning from the great presenters."

We expect affordability to be a major focus at this year's event. It is a central tenet of the DoD's Better Buying Power initiatives, and budget projections indicate it will continue to be important as the nation works its way out of the recession. This suggests that research with a focus on affordability will be of great interest to the DoD leadership in the year to come. Whether you're a practitioner or scholar, we invite you to participate in that research.

We gratefully acknowledge the ongoing support and leadership of our sponsors, whose foresight and vision have assured the continuing success of the ARP:

- Office of the Under Secretary of Defense (Acquisition, Technology, & Logistics)
- Director, Acquisition Career Management, ASN (RD&A)
- Program Executive Officer, SHIPS
- Commander, Naval Sea Systems Command
- Program Executive Officer, Integrated Warfare Systems
- Army Contracting Command, U.S. Army Materiel Command
- Office of the Assistant Secretary of the Air Force (Acquisition)

- Office of the Assistant Secretary of the Army (Acquisition, Logistics, & Technology)
- Deputy Director, Acquisition Career Management, U.S. Army
- Office of Procurement and Assistance Management Headquarters, Department of Energy
- Director, Defense Security Cooperation Agency
- Deputy Assistant Secretary of the Navy, Research, Development, Test & Evaluation
- Program Executive Officer, Tactical Aircraft
- Director, Office of Small Business Programs, Department of the Navy
- Director, Office of Acquisition Resources and Analysis (ARA)
- Deputy Assistant Secretary of the Navy, Acquisition & Procurement
- Director of Open Architecture, DASN (RDT&E)
- Program Executive Officer, Littoral Combat Ships

We also thank the Naval Postgraduate School Foundation and acknowledge its generous contributions in support of this symposium.

James B. Greene Jr.
Rear Admiral, U.S. Navy (Ret.)

Keith F. Snider, PhD
Associate Professor

Panel 10. Acquisition Strategies for Software-Intensive Systems

Wednesday, May 16, 2012	
3:30 p.m. – 5:00 p.m.	**Chair: Mr. Pat Sullivan,** Executive Director, Program Executive Officer, Command, Control, Communications and Intelligence ***Cross-Program Weapons System Software Acquisition Can Save Billions*** Rick Brennan, *Operational Systems Inc.* ***An Innovative Approach to Lower the Risk of Software Intensive Development Programs*** Jeff Dunlap, *BAE Systems* ***Software Strategy for the Defense Enterprise*** John Robert, *Software Engineering Institute*

Pat Sullivan—Mr. Sullivan is the executive director for the Program Executive Office for Command, Control, Communications, Computers and Intelligence (PEO C4I), San Diego, CA. Mr. Sullivan is responsible for integrating, executing and delivering capability in a $2.5 billion portfolio supporting information needs for naval, joint, and coalition warfighters. Mr. Sullivan received a bachelor's degree in electrical and computer engineering from the University of California, San Diego (UCSD) in 1989 and continued at the university to earn his master's degree in electrical engineering and applied physics in 1991.

Mr. Sullivan began his government career at the Naval Ocean Systems Center a predecessor of the Space and Naval Warfare Systems Center Pacific (SSC-Pacific). From 1991 to 1996, he was a project manager for the Design and Development Branch, working with the Defense Advanced Research Projects Agency Electronics Technology Office to develop new initiatives in the area of advanced electronic packaging. From September 1996 to January 2000, Mr. Sullivan was a project manager for the Integrated Circuit Research and Fabrication Branch, responsible for developing, managing, and performing as principal investigator for several advanced microelectronic research and development projects.

In January 2000, Mr. Sullivan assumed responsibilities as the head of the Integrated Circuit Research and Fabrication Branch, where he focused on microelectronic technology development for the strategic space and intelligence communities. From August 2002 through June 2006, Mr. Sullivan led the Joint and National Systems Division, supplying advanced technology to the intelligence and special operations communities. In March 2006, Mr. Sullivan was selected to lead the Intelligence, Surveillance and Reconnaissance Department and entered the Senior Executive Service later that year. His responsibilities in this position at SSC-Pacific included managing a broad set of programs to develop capabilities in the areas of maritime surveillance and ocean systems, joint and national information systems, intelligence systems, signal exploitation and cryptologic systems, and systems to support information operations and battlespace awareness. He also served as SPAWAR Engineering's National Competency Lead for ISR and Information Operations. He assumed his current position with PEO C4I in October 2010. Mr. Sullivan is a member of the UCSD Electrical and Computer Engineering Advisory Board, the National Defense Industrial Association, Armed Forces Communications and Electronics Association, and the Acquisition Professional Community.

An Innovative Approach to Lower the Risk of Software-Intensive Development Programs

Jeff Dunlap—CAPT (Ret.) Dunlap has 25 years of government program management and operational experience, culminating with the assignment of deputy program manager with the DoD's largest software-intensive radio development program, Joint Tactical Radio Systems. Dunlap is currently the director for Navy C4ISR with BAE Systems, Intelligence and Security in San Diego, CA. [jeff.dunlap@baesystems.com]

Abstract

Since 1973, nearly 80% of DoD ACAT I programs have experienced cost overruns, coupled with a four-out-of-five chance of not fielding capability to the warfighter on time. With the DoD acquisition reforms of the past two decades, the probability of program success (PoPS) rate is improving. To continue improving PoPS, program management tools and techniques need to develop and become institutionalized to monitor software-intensive-based capability, control, and logic development efforts.

A lack of sufficient tools to monitor software development costs and performance to the integrated baseline drives uncertainty and risk. The earned value management system (EVMS) approach lacks meaningful measures for software-intensive development programs, and "you can't control what you can't measure" (DeMarco, 1986, p. 58).

Using commercially available (often freeware) tools, a robust set of automated managers can measure near real-time progress and identify trouble areas early in the software development process to allow meaningful correction to occur. This paper explores agile sprint development and continuous integration and test best practices, and the potential for an innovative approach of intertwining the two to reduce risk and increase PoPS.

Introduction

Today's modern warfare systems have become dependent on computers to reduce the speed to bang and to shorten the observe, orient, decide, and act engagement loops. Computers have increased the survivability of the friendly forces and have facilitated advanced tactics and procedures to provide substantial advantage during conflict. Since warfare systems must constantly evolve to maintain tactical advantages, computers have provided the ideal canvas on which software code can be re-crafted to take advantage of Moore's law and enable more capable and cheaper weapons and IT systems.

As warfare systems evolved to take advantage of computer systems, the ability to monitor the software development progress became less of a mathematical exercise, but more of the black art business of financial management wizards. The earned value measurement system (EVMS), which worked so well for measuring sheets of metal welded in a week (the physical arts), has become irrelevant for managing software-intensive development cost and progress.

The EVMS provides a rearview mirror of the past, which usually lags reality by more than 45 days when presented to the program manager (PM). Numerous reviews of the EVMS data and interpretation of trends from these events of the past provide little value and often lead to bad practices by industry. The physical arts had no issue with this delay, since one could always observe the beads of weld being laid and get a real-time gut check (measurement metric) on progress. On the contrary, with software development, progress is often reported in lines of software code produced in a day, which provides little value as a measurement metric. Performance difficulties are often masked by false indications of progress, which lead to cost and schedule deviations. The EVMS in a software-intensive development program is a non-productive process that provides little value to the PM.

There are three things that must occur to increase the probability of program success and provide increased visibility during a software-intensive development:

1. A method of defining and executing capability development that directly supports software-intensive programs,
2. A reduction in uncertainty by measuring the actual progress and employing automated testing while the software is being developed, and
3. The ability of the customer to adapt their business processes to encourage innovative agile systems development accompanied with changes in the acquisition process.

The agile methodology examined in this paper provides insight into the schedule tailoring and upfront systems engineering required to develop capability in iterative sprints. A shift in thinking must occur that encourages failures to be exposed early and without consequence. An assertion is made that any agile methodology implemented is only as successful as the managers' and leaders' willingness to change their processes. The ability to track progress and monitor costs becomes more tactical with agile methodologies and the government oversight processes must evolve to capitalize on this visibility.

Using continuous integration and testing (CI&T) best practices and tools not only benefits the software coder and tester, but also provides real-time status on progress and the goodness of the integrated build. The need to hit the "just trust me" button of traditional software development is eliminated by employing CI&T outputs in a non-traditional manner.

The big-bang theory of providing capability to integrated testing (typical DoD 5000) is inherently flawed for software-intensive IT systems and requires tailored change to the core acquisition framework. The DoD is examining agile business processes to improve efficiency and effectiveness in IT acquisition, as mandated by Congress in Section 804 of the National Defense Authorization Act for Fiscal Year 2010 (2009). A concept of acquisition tailoring for IT acquisitions to enable agile capability delivery is reviewed with additional recommendations.

By themselves, each of the changes can be successful, but the coupling effect of all three has the ability to achieve and innovative a step forward to increase the probability of program success.

Methods

At BAE Systems, Intelligence and Security, several engineering and programmatic changes have been instituted that provide new methods to lower the risk of software-intensive development. By comparing and contrasting these methods to traditional DoD 5000 processes, an innovative recommendation is offered to apply these solutions to achieve a greater programmatic probably of success.

The first recommended risk reduction component is the implementation of agile methodologies in capability development. Agile is customer focused and requires change in the government's oversight process. If you consider the DoD 5000 systems engineering "big V," agile looks like lots of "small v's." Roadmaps and architectures align agile small-v increments into larger big-V capabilities. There are several agile methodologies and each has advantages that are dependent on the type of capability being developed. The scrum agile method is one of the methods employed at BAE Systems and will be overviewed in the following section.

The second recommended risk reduction component is continuous integration and testing (CI&T). CI&T is a best practice that really focuses on quality in testing and provides an end-to-end test process that is independent of the software development process method (i.e., waterfall, agile, spiral, etc.). Continuous integration is simply an advance in the evolution of integrating software and employing automated build tools. By adding automated testing to the integrated builds, problems are discovered early with automated feedback to the software coder and tester. CI&T is not a simple plug-and-play capability, but rather a consortium of commercial tools that are specifically picked by the engineering development team. With sufficient upfront considerations and tool integration, CI&T has shown significant return on investment and decreased program risk.

The third recommended risk reduction component is the hardest of all to implement due in part to the inertia of the current DoD acquisition process, resistance to change at the government oversight levels, and regulatory reporting requirements of the PM. The challenges are well known and the benefits of successful reform are critical in a constrained budget environment. Office of the Secretary of Defense (OSD; 2010) efforts for tailoring the process for the acquisition of information technology are presented and several innovative programmatic ideals discussed.

Discussion

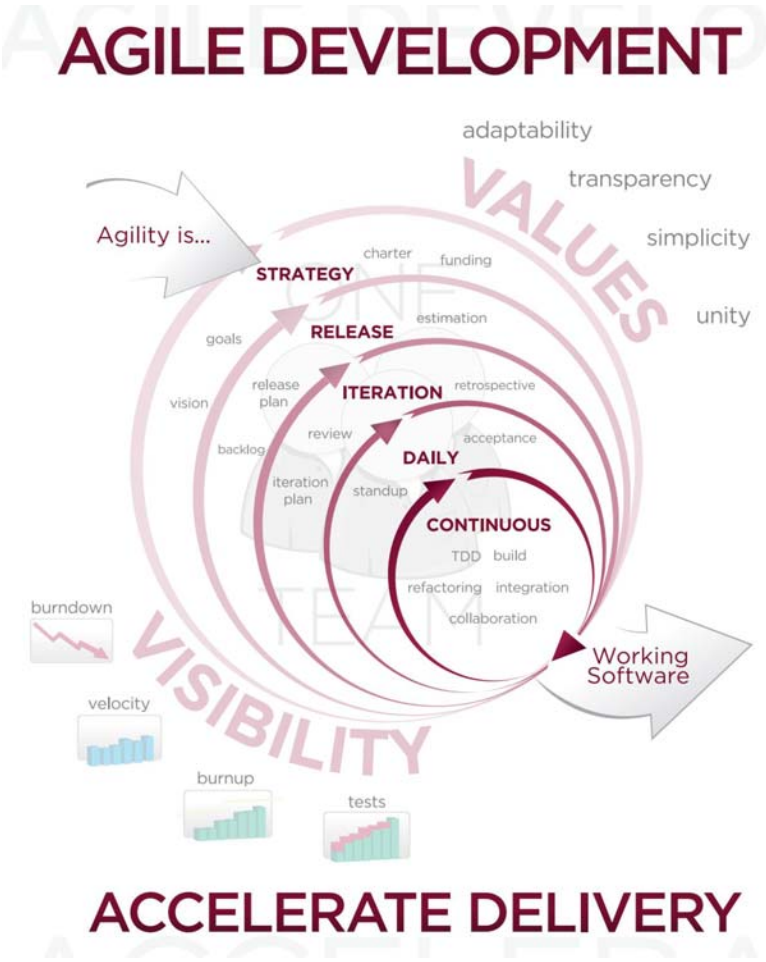


Figure 1. The Agile Development Process

Agile software development does not follow a typical DoD development process. Figure 1 has agility presented as a non-liner and continuous process of collaboration and visibility into the product being developed. The software development methods are iterative and incremental, where requirements and solutions evolve through collaboration between self-organizing, cross-functional teams. Agile development promotes adaptive planning, evolutionary development and delivery, and a time-constrained iterative approach. It also encourages rapid and flexible response to change.

Scrum is an agile methodology used as an example that provides a structured approach to delivering capabilities in increments based on the program management roadmaps and architectures that align into larger capabilities.

Key components of the scrum process include the following:

- Prior to each sprint customer meeting, a prioritized list of capabilities (or Epics) is developed.
- Customer-prioritized lists of capabilities are broken down into stories by leadership (including top-down “story point” estimations) resulting in a planning spreadsheet.
- Once the desired capability is set for an iteration, no external addition of work can be added.
- Teams are self-directed and self-organized.
- Daily stand-up meetings are held that include questions about yesterday’s progress, plans for the day, and difficulties encountered.
- The process is usually completed in 30-day iterations.
- Demos to external stakeholders happen at the end of each iteration.
- Sprint retrospective (internal feedback) occurs on the last day.

Scrum Roles

- Customer/Product Owner
    - Prioritizes backlog
    - Chooses goals for next sprint (iteration)
    - Reviews the system at the end of each sprint with a demo of the capability
- Scrum Master
    - A developer
    - Organizes the process
    - Removes obstacles
    - Mediates between management and the team
    - Conducts daily scrum (stand-up) and sprint review (demo)
- Team
    - Organizes itself to perform the work and deliver business value
- All others
    - Can observe but not interfere

Scrum Work Products

- Product Backlog
    - Used to determine work for next sprint
    - A prioritized list of everything needed or wanted for the entire product

    - Often written in the form of user stories
    - Has rough estimates associated with each item

- Sprint Backlog
    - List of tasks to be completed in a sprint
    - Tasks created by breaking down the stories during the planning meeting
    - Has estimates (often in hours) associated with them

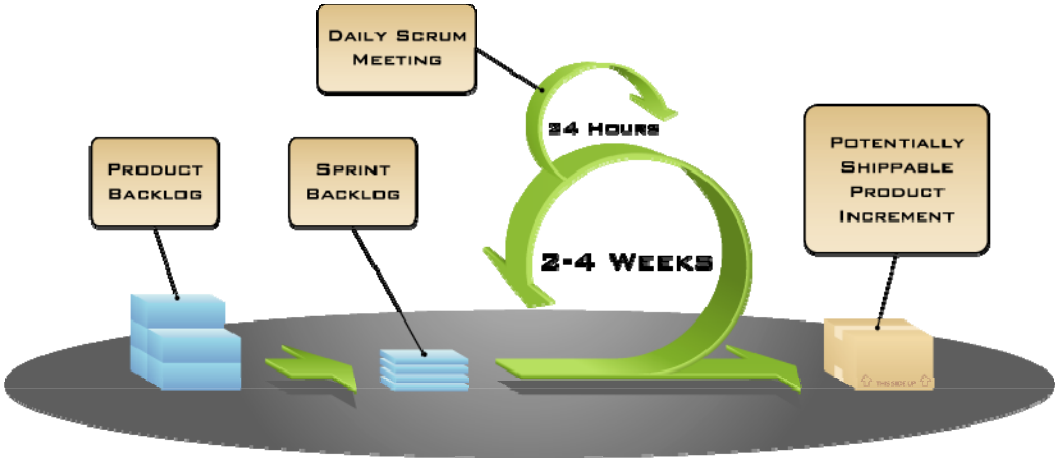


Figure 2. The Scrum Process
(Larman, 2003)

Scrum Events

- Sprint Planning Meeting
    - Team selects stories it believes it can achieve in the next sprint
    - Breaks stories down into tasks and provides estimates
- Sprint
    - 30 days
    - Where development work occurs (design, implement, test, document)
- Daily Scrum (stand-up)—15 minutes
    - Team members share progress with other team members
    - Three questions
        - What did you do yesterday?
        - What will you do today?
        - Do you have any roadblocks?
    - Anyone may attend, but only the team may speak
- Sprint Review
    - Product owner reviews product and provides feedback
- Sprint Retrospective
    - Team reviews what went well and what went poorly
    - Find areas for improvement

Scrum Product Backlog

- Contains stories
    - Independent: Should be independent from other stories
    - Negotiable: Should have room to negotiate (starting point, not a contract)
    - Valuable: Should communicate value to a user or customer
    - Estimate-able: If it’s too complex to estimate, break it up into multiple pieces.
    - Small: Incremental functionality
    - Testable: To validate story was correctly implemented
- Prioritized by product owner / customer (or surrogate)
- Daily update on team progress
    - Each team member estimates time remaining for each task in work

The upfront planning needed to prepare the capability roadmaps and architectures to align agile increments into larger capabilities is paramount prior to starting the journey. The agile development process has several desirable factors and metrics that can be used to reduce risk and uncertainty within the project. The entire project baseline is developed in advance and, based on the complexity of the product desired, may contain a large number of scrum events. The value of 30-day iterations with demonstrations has two important risk-reduction properties:

1. The development has real-time measurable progress via the daily scrum meetings. Problems are addressed early with the ability to surge support into difficult areas as needed.
2. Cost is captured with validated schedule progress. Performance is observed at the end of the agile incremental sprint by the customer.

Continuous Integration & Testing (CI&T) Best Practices

In his popular “Continuous Integration” article, Martin Fowler (2006) describes CI as “a software development practice where members of a team integrate their work frequently, usually each person integrates at least daily—leading to multiple integrations per day. Each integration is verified by an automated build (including test) to detect integration errors as quickly as possible.” CI is simply an advance in the evolution of integrating software and must be continuous and automated at every stage.

Effective application of CI practices can provide greater confidence in producing a software product. The benefits of implementing CI are reduced risks, reduced repetitive manual processes, generation of deployable software at any time, better project visibility, and, ultimately, greater confidence in the software product.

Table 1. Benefits of Continuous Integration

Reduce risks	Defects are detected and fixed sooner The health of software is measurable Assumptions are reduced—building in a consistent, clean environment
Reduce repetitive manual processes	The process runs the same way every time
Generate deployable software at any time and at any place	From an outside perspective, the most obvious benefit of CI
Enable better project visibility	Effective decisions Notice trends

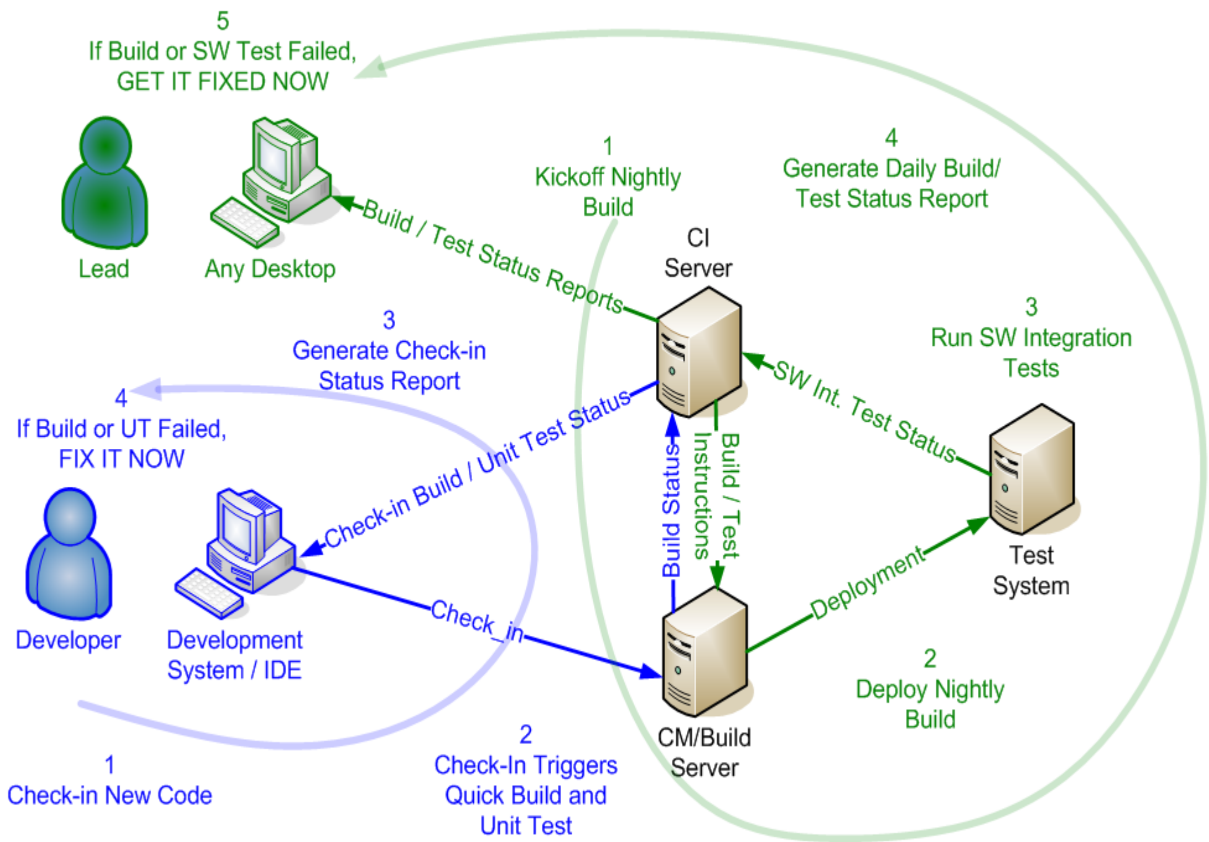
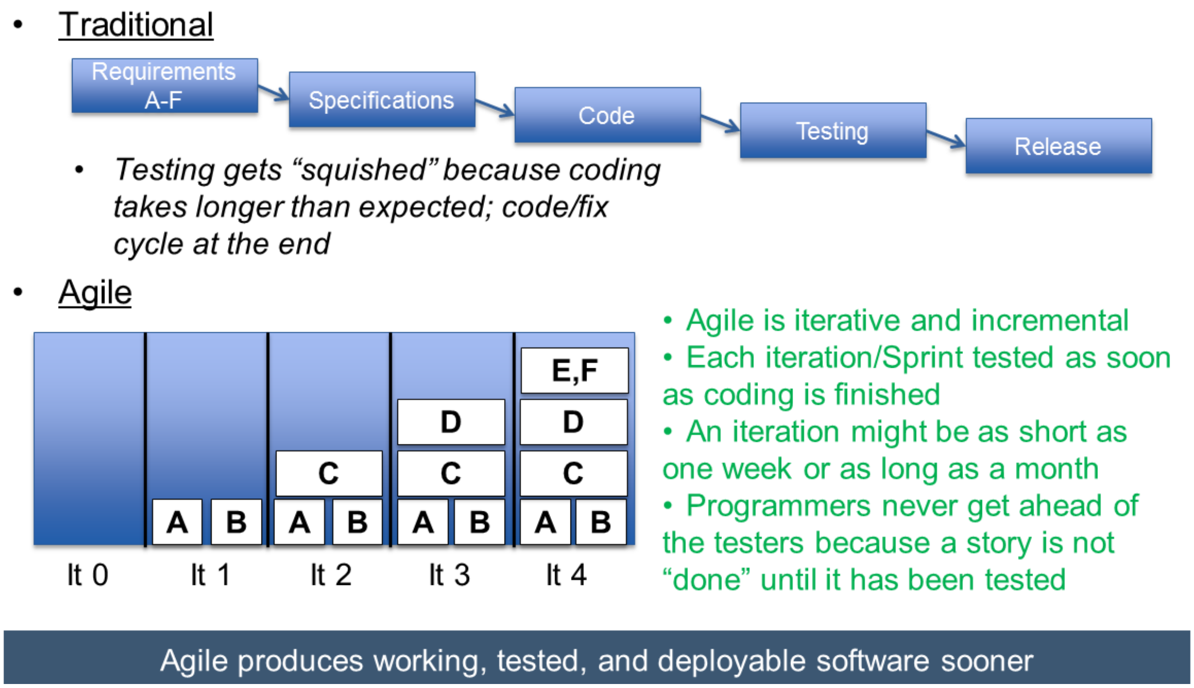


Figure 3. Components of a Continuous Integration System

The typical process includes the following elements:

- All developers run private builds on their own workstations before committing their code to the version-control repository to ensure that their changes work.
- Developers commit their code to a version-control repository *at least once a day.*
- Integration builds occur several times a day on a separate build machine.
- 100% of tests must pass for every build.
- A product is generated that can be functionally tested.
- Fixing broken builds is of the highest priority.
- Some developers review reports generated by the build, such as coding standards and dependency analysis reports, to seek areas for improvement.



- Traditional

Requirements A-F → Specifications → Code → Testing → Release

- *Testing gets "squished" because coding takes longer than expected; code/fix cycle at the end*

- Agile

- Agile is iterative and incremental
- Each iteration/Sprint tested as soon as coding is finished
- An iteration might be as short as one week or as long as a month
- Programmers never get ahead of the testers because a story is not "done" until it has been tested

Agile produces working, tested, and deployable software sooner

Figure 4. Traditional Versus Agile Development

What is agile testing?

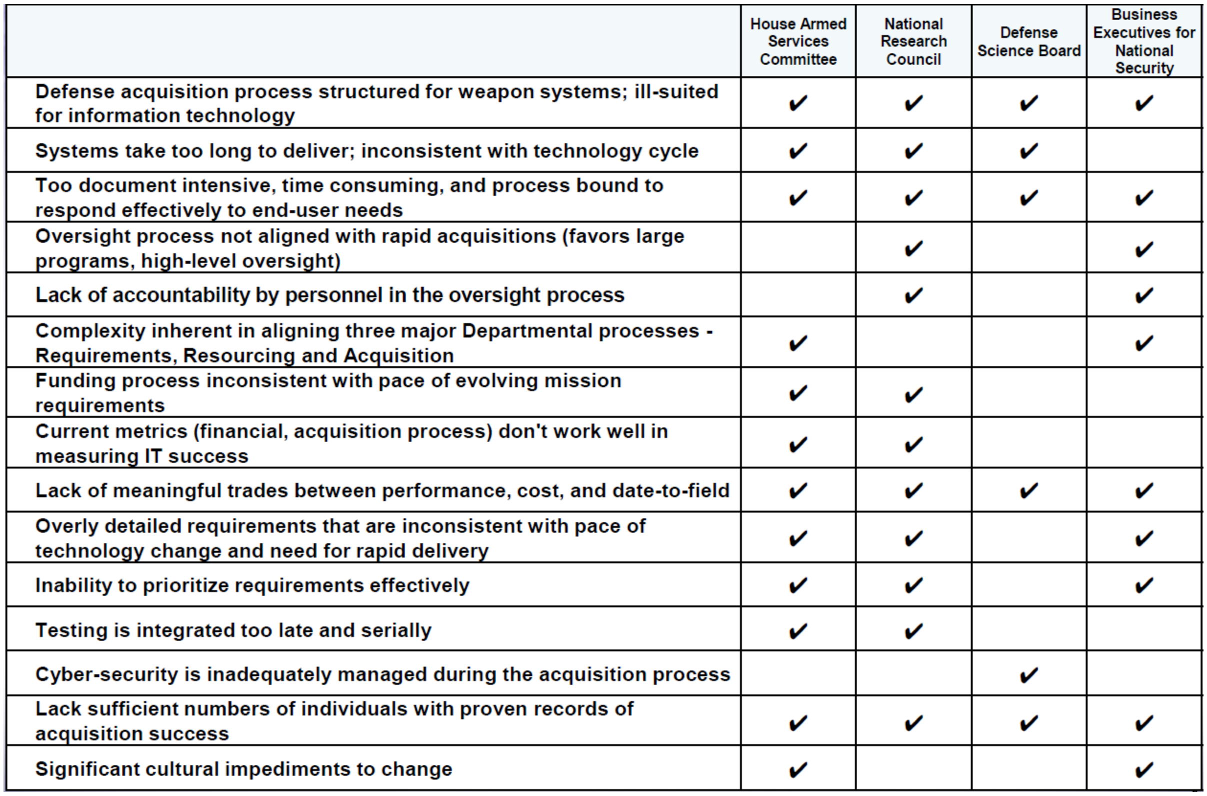
- Testers do more than perform "testing tasks."
- Each agile team member is focused on delivering a high-quality product that provides business value.
- Agile testers work to ensure that their team delivers the quality their customers need.
- Agile programmers use test-driven development:
  - Programmers write a test for a tiny bit of functionality before writing more code.
  - Programmers also write code integration tests to make sure the small units of code work together as intended.

To incorporate agile testing successfully in a DoD program, the government's oversight and involvement becomes critical at the incremental iteration level, as there is generally no big bang to observe.

The degree of test automation, and the timing of its employment, must be based on the investment required and the return desired. The goal is to achieve 100% automation at the unit level where the best return on investment occurs.

DoD Acquisition Processes

The need for the DoD to change and to keep pace with technology changes and budget challenges has driven mandated congressional changes to acquisition processes, as shown in Figure 5.



	House Armed Services Committee	National Research Council	Defense Science Board	Business Executives for National Security
Defense acquisition process structured for weapon systems; ill-suited for information technology	✔	✔	✔	✔
Systems take too long to deliver; inconsistent with technology cycle	✔	✔	✔	
Too document intensive, time consuming, and process bound to respond effectively to end-user needs	✔	✔	✔	✔
Oversight process not aligned with rapid acquisitions (favors large programs, high-level oversight)		✔		✔
Lack of accountability by personnel in the oversight process		✔		✔
Complexity inherent in aligning three major Departmental processes - Requirements, Resourcing and Acquisition	✔			✔
Funding process inconsistent with pace of evolving mission requirements	✔	✔		
Current metrics (financial, acquisition process) don't work well in measuring IT success	✔	✔		
Lack of meaningful trades between performance, cost, and date-to-field	✔	✔	✔	✔
Overly detailed requirements that are inconsistent with pace of technology change and need for rapid delivery	✔	✔		✔
Inability to prioritize requirements effectively	✔	✔		✔
Testing is integrated too late and serially	✔	✔		
Cyber-security is inadequately managed during the acquisition process			✔	
Lack sufficient numbers of individuals with proven records of acquisition success	✔	✔	✔	✔
Significant cultural impediments to change	✔			✔

Figure 5. Assessment of Current DoD Information Technology Acquisition Challenges
(Pontius, 2012)

Section 804 of the National Defencse Authorization Act of FY2010 (2009) requires a new IT acquisition process. The IT acquisition reform's overarching principles provide a simplified, tailorable approach for delivering IT capability that favors mature technology, emphasizes the enterprise, eliminates redundancy, and includes the following:

- Early and frequent delivery—be responsive to the users' needs,
- Incremental and iterative development and testing,
- Rationalized requirements—balance user needs with constraints,
- Flexible/tailored processes—customize to IT category, and
- Knowledgeable and experience IT workforce—understands IT's uniqueness.

To enable agile IT software-intensive capability delivery, the Office of the Secretary of Defense is reviewing several key changes:

- Utilize streamlined contracting processes to leverage existing contract vehicles for rapid Task/Delivery Order execution;
- Leverage common infrastructure platforms, standards, and interfaces;
- Integrate test and evaluation and certifications during development, leveraging common test infrastructure and automated tools;
- Develop roadmaps and architectures to align agile increments into larger capabilities;
- Implement agile project management approaches to include small, dynamic, and empowered teams;

- Involve users actively to prioritize requirements and provide responsive feedback during development; and
- Deliver usable larger capabilities (built upon agile increments) every 6–12 months.

Recommendations

Key Software Intensive Development Program Items to Consider

How to Ensure Affordability and Control Growth

By implementing agile capability development and CI&T, the results produced are timely delivery of effective and efficient capabilities. Sprints are cost and schedule constrained and the metrics provided by CI&T allow real-time progress to be monitored. With an emphasis on affordability and short program timelines, agile methods like scrum reveal cost and schedule issues early, and corrective actions can be taken to recover or re-engineer the sprint.

How to Incentivize Productivity and Innovation in Industry

With the desire to increase the use of Fixed Priced Incentive Fee (FPIF) contract types, industry is challenged to innovate new ways of applying program management and systems development tools to control cost and schedule since FPIF shifted the risk to the contractor. By eliminating the costly and burdensome EVMS requirements and focusing on innovative methodologies of combining the right tools needed to maintain program cost and schedule, visibility will result in a win-win outcome. Innovative thought, such as combining two entirely different processes (e.g., scrum and CI&T) and eliminating non-valued reporting systems, can spark productivity.

How to Promote Real Competition

The ability to innovate and implement agile processes requires investment by industry, both in training and tools. If the request for proposal and the subsequent source selection criteria do not rate agile and other innovative process sufficiently high, then the change needed will not occur. The current trend of technically acceptable / lowest cost competitions focuses on the wrong set of metrics to promote real innovation and competition. Agile development provides opportunities for more frequent competition and encourages the use of open systems architectures. Deliberate thought should be given to what is the desired result from the competition. Competition for its own sake does not always bring value to the government and can sometimes lead to disruption and cost growth.

Reduce Non-Productive Processes and Bureaucracy

Employing agile and CI&T processes provides streamlined test and certification, with both the customer and external test observer integrated in the 30-day demo and delivery process. Agile incremental sprints are aligned into larger capabilities testing with lower risk of failback. The effect of the elimination of the EVMS from agile software-intensive systems is substantial and can offset the additional engineering and planning cost incurred for each agile sprint. With the real-time reporting of cost, schedule, and observed performance of each sprint, the EVMS becomes non-productive to the program manager.

References

DeMarco, T. (1986). *Controlling software projects: Management, measurement, and estimates*.

Fowler, M. (2006, May 1). *Continuous integration*.

Larman, C. (2003, August 21). *Agile and iterative development: A manager's guide*.

National Defense Authorization Act for Fiscal Year 2010, H.R. 2647, 111 Cong. (2009).

Office of the Secretary of Defense. (2010, November). *A new approach for delivering information technology capabilities in the Department of Defense* [Report to Congress].

Pontius, R. W. (2012, January 20). *Acquisition of information technology: Improving efficiency and effectiveness in information technology acquisition in the Department of Defense* [Presentation slides from the Director, Command and Control Programs & Policy (OSD)].

ACQUISITION RESEARCH PROGRAM
GRADUATE SCHOOL OF BUSINESS & PUBLIC POLICY
NAVAL POSTGRADUATE SCHOOL
555 DYER ROAD, INGERSOLL HALL
MONTEREY, CA 93943

www.acquisitionresearch.net

BAE SYSTEMS

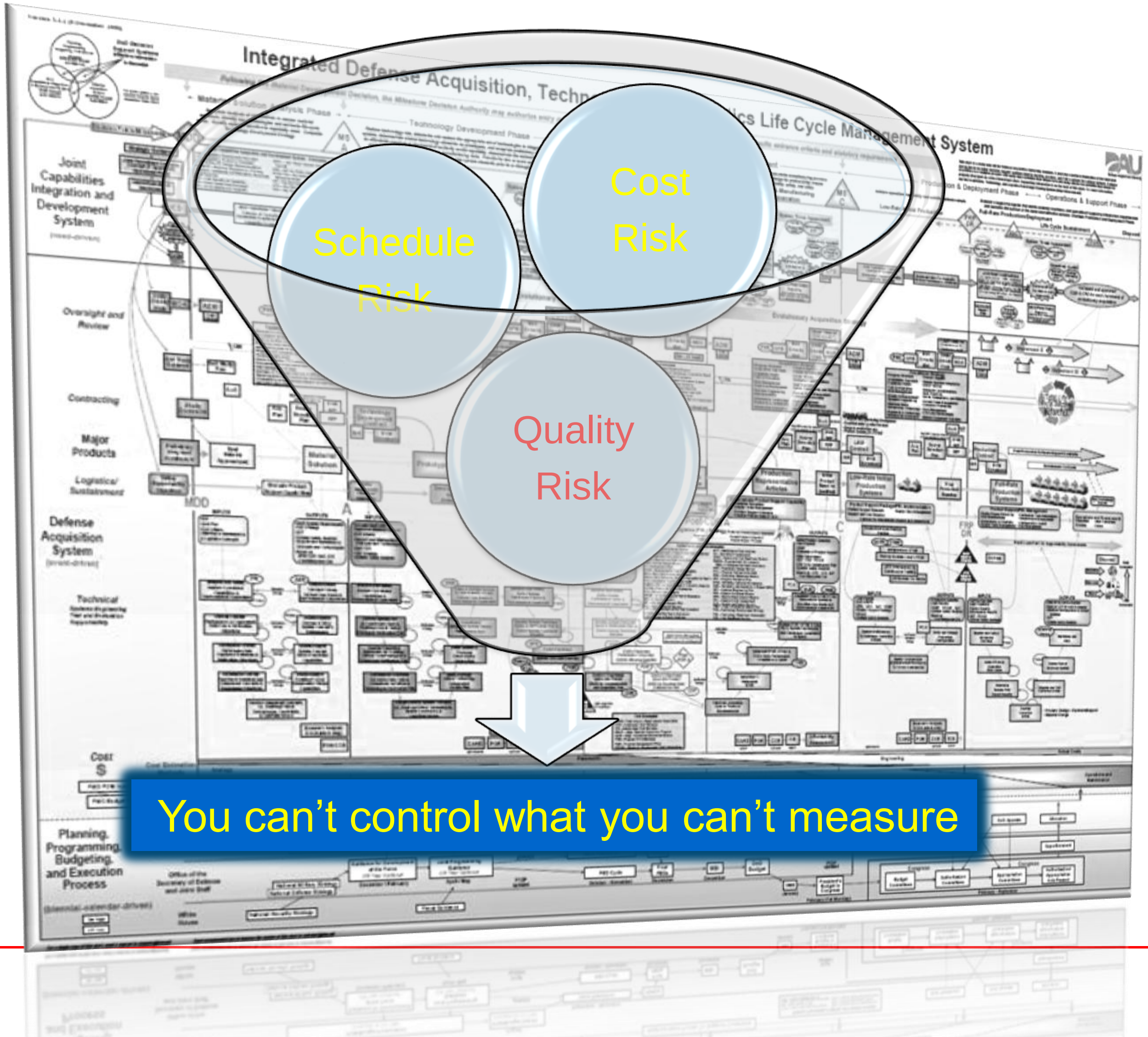
An Innovative Approach to Lower the Risk of Software Intensive Development Programs

Jeff Dunlap

Director Navy C4ISR

Jeff.dunlap@baesystems.com

Assertion: Less risk = greater probability of success


Integrated Defense Acquisition, Techn... ics Life Cycle Management System
Joint Capabilities Integration and Development System
Oversight and Review
Contracting
Major Products
Logistics/ Sustainment
Defense Acquisition System
Technical
Cost
Planning, Programming, Budgeting, and Execution Process

Schedule Risk

Cost Risk

Quality Risk

You can’t control what you can’t measure

Elements needed to control risk

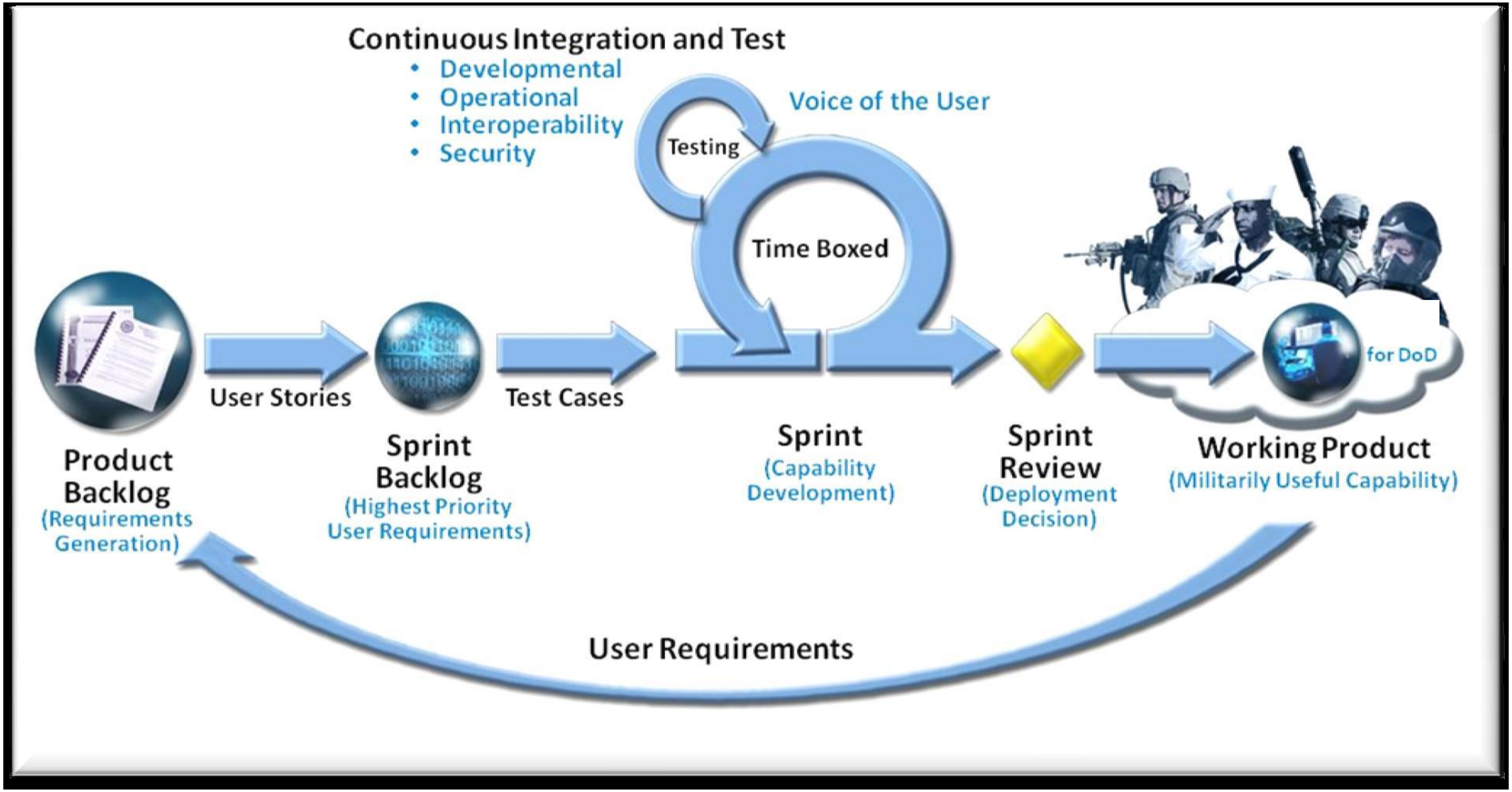
☑ constant measure of schedule achieved

☑ constant measure of actual cost incurred

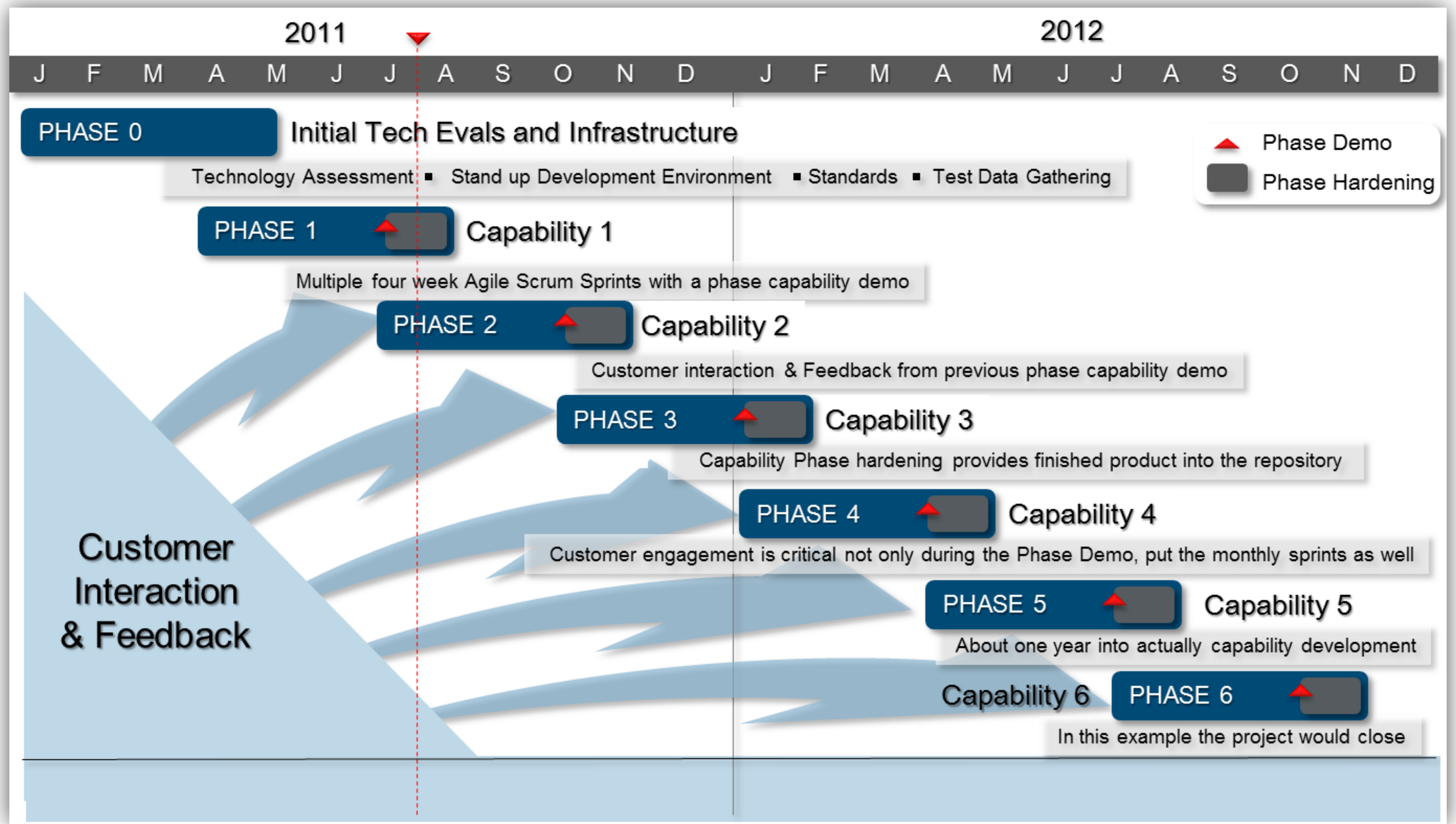
☑ constant measure of quality tested

☑ Agile Earned Value

Agile Development



Industry example of a System Capability Roadmap



2011

2012

J F M A M J J A S O N D J F M A M J J A S O N D

PHASE 0 Initial Tech Evals and Infrastructure

Technology Assessment ▪ Stand up Development Environment ▪ Standards ▪ Test Data Gathering

▲ Phase Demo

▨ Phase Hardening

PHASE 1 Capability 1

Multiple four week Agile Scrum Sprints with a phase capability demo

PHASE 2 Capability 2

Customer interaction & Feedback from previous phase capability demo

PHASE 3 Capability 3

Capability Phase hardening provides finished product into the repository

PHASE 4 Capability 4

Customer engagement is critical not only during the Phase Demo, put the monthly sprints as well

PHASE 5 Capability 5

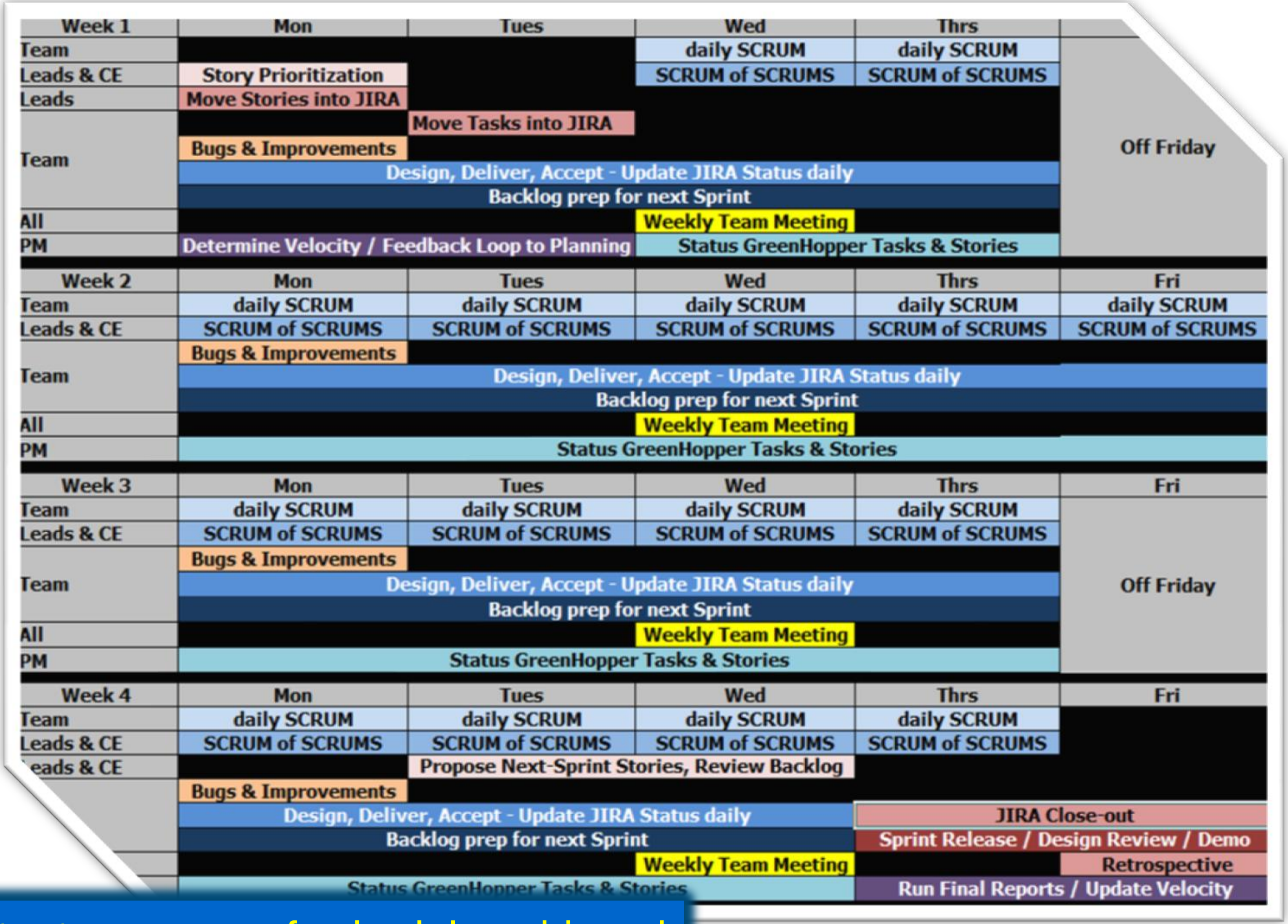
About one year into actually capability development

Capability 6 PHASE 6

In this example the project would close

Customer Interaction & Feedback

Agile four week Scrum Sprint within a capability phase



Week 1	Mon	Tues	Wed	Thrs	
Team			daily SCRUM	daily SCRUM	Off Friday
Leads & CE	Story Prioritization		SCRUM of SCRUMS	SCRUM of SCRUMS	
Leads	Move Stories into JIRA				
Team		Move Tasks into JIRA			
	Bugs & Improvements				
	Design, Deliver, Accept - Update JIRA Status daily				
	Backlog prep for next Sprint				
All			Weekly Team Meeting		
PM	Determine Velocity / Feedback Loop to Planning		Status GreenHopper Tasks & Stories		
Week 2	**Mon**	**Tues**	**Wed**	**Thrs**	**Fri**
Team	daily SCRUM	daily SCRUM	daily SCRUM	daily SCRUM	daily SCRUM
Leads & CE	SCRUM of SCRUMS	SCRUM of SCRUMS	SCRUM of SCRUMS	SCRUM of SCRUMS	SCRUM of SCRUMS
Team	Bugs & Improvements				
	Design, Deliver, Accept - Update JIRA Status daily				
	Backlog prep for next Sprint				
All			Weekly Team Meeting		
PM	Status GreenHopper Tasks & Stories				
Week 3	**Mon**	**Tues**	**Wed**	**Thrs**	**Fri**
Team	daily SCRUM	daily SCRUM	daily SCRUM	daily SCRUM	Off Friday
Leads & CE	SCRUM of SCRUMS	SCRUM of SCRUMS	SCRUM of SCRUMS	SCRUM of SCRUMS	
Team	Bugs & Improvements				
	Design, Deliver, Accept - Update JIRA Status daily				
	Backlog prep for next Sprint				
All			Weekly Team Meeting		
PM	Status GreenHopper Tasks & Stories				
Week 4	**Mon**	**Tues**	**Wed**	**Thrs**	**Fri**
Team	daily SCRUM	daily SCRUM	daily SCRUM	daily SCRUM	
Leads & CE	SCRUM of SCRUMS	SCRUM of SCRUMS	SCRUM of SCRUMS	SCRUM of SCRUMS	
eads & CE		Propose Next-Sprint Stories, Review Backlog			
	Bugs & Improvements				
	Design, Deliver, Accept - Update JIRA Status daily			JIRA Close-out	
	Backlog prep for next Sprint			Sprint Release / Design Review / Demo	
			Weekly Team Meeting		Retrospective
	Status GreenHopper Tasks & Stories			Run Final Reports / Update Velocity	

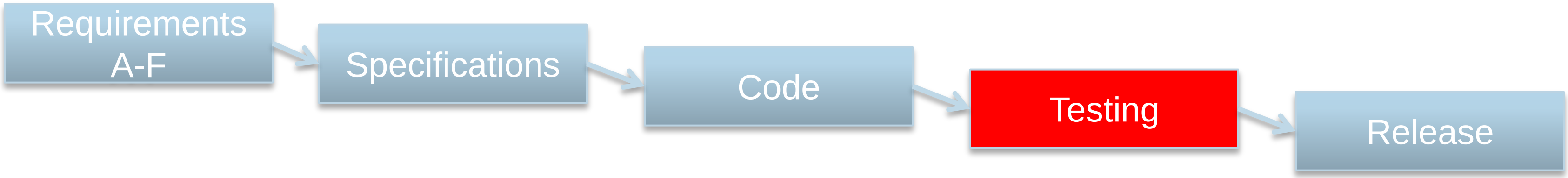
Constant measure of schedule achieved

Constant measure of actual costs incurred

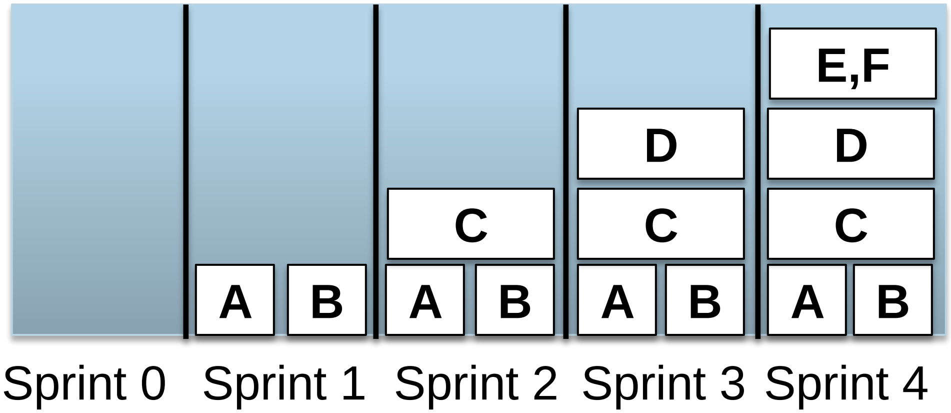
Measure of quality tested

Traditional vs. Agile Testing

Traditional

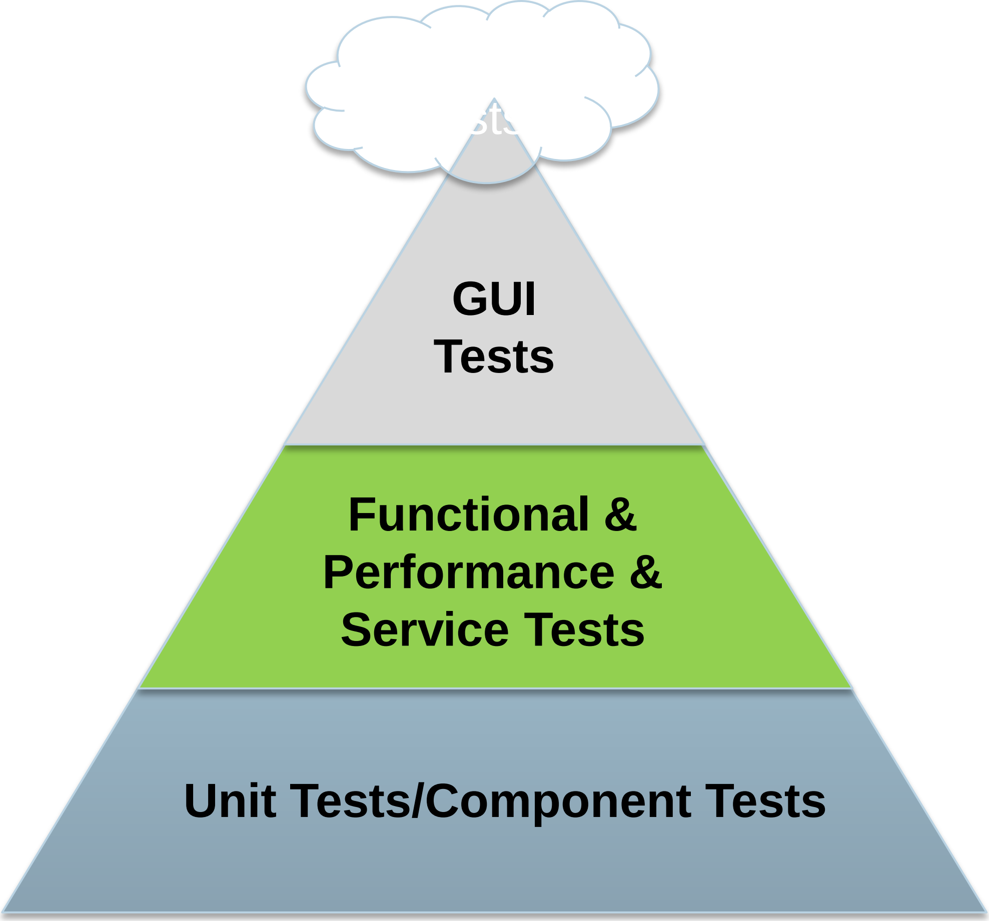


Agile



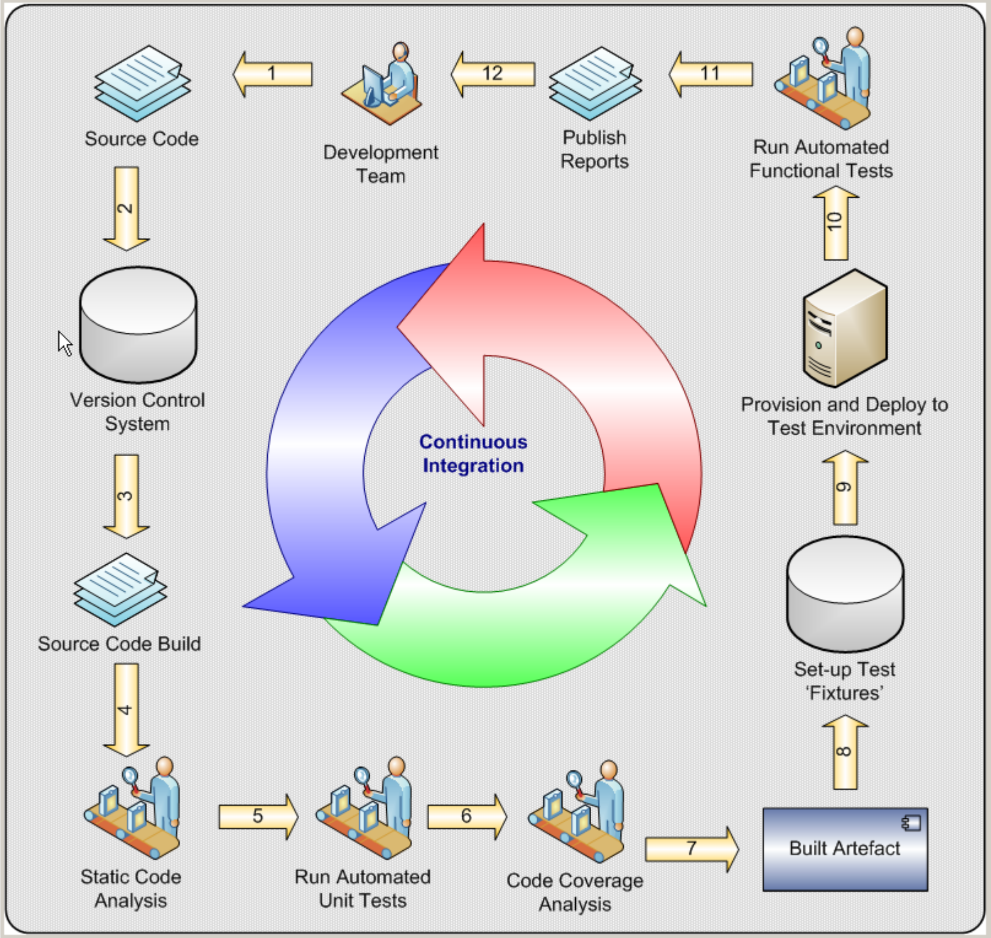
Agile produces working, tested, and deployable software sooner

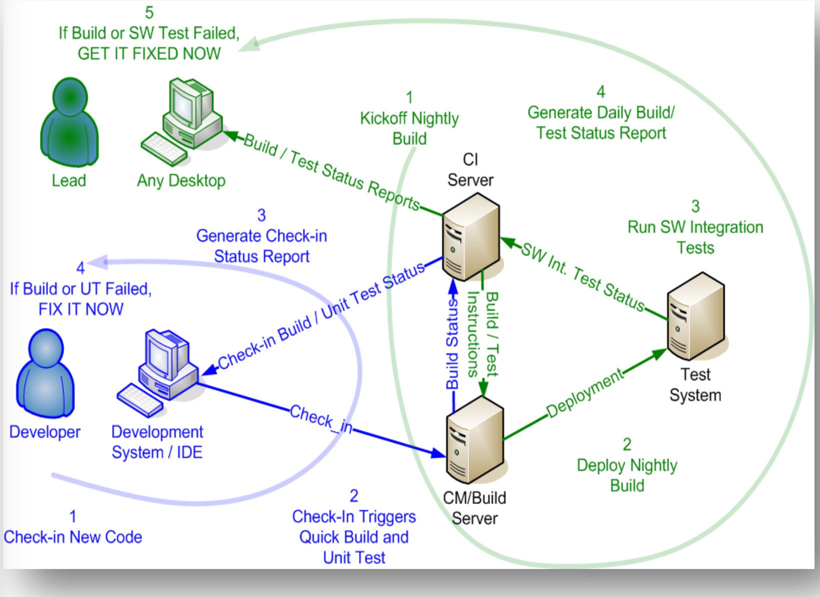
Test Automation Pyramid


sts
GUI Tests
Functional & Performance & Service Tests
Unit Tests/Component Tests

Goal is to achieve 100% automation at the Unit Level – where the best ROI occurs

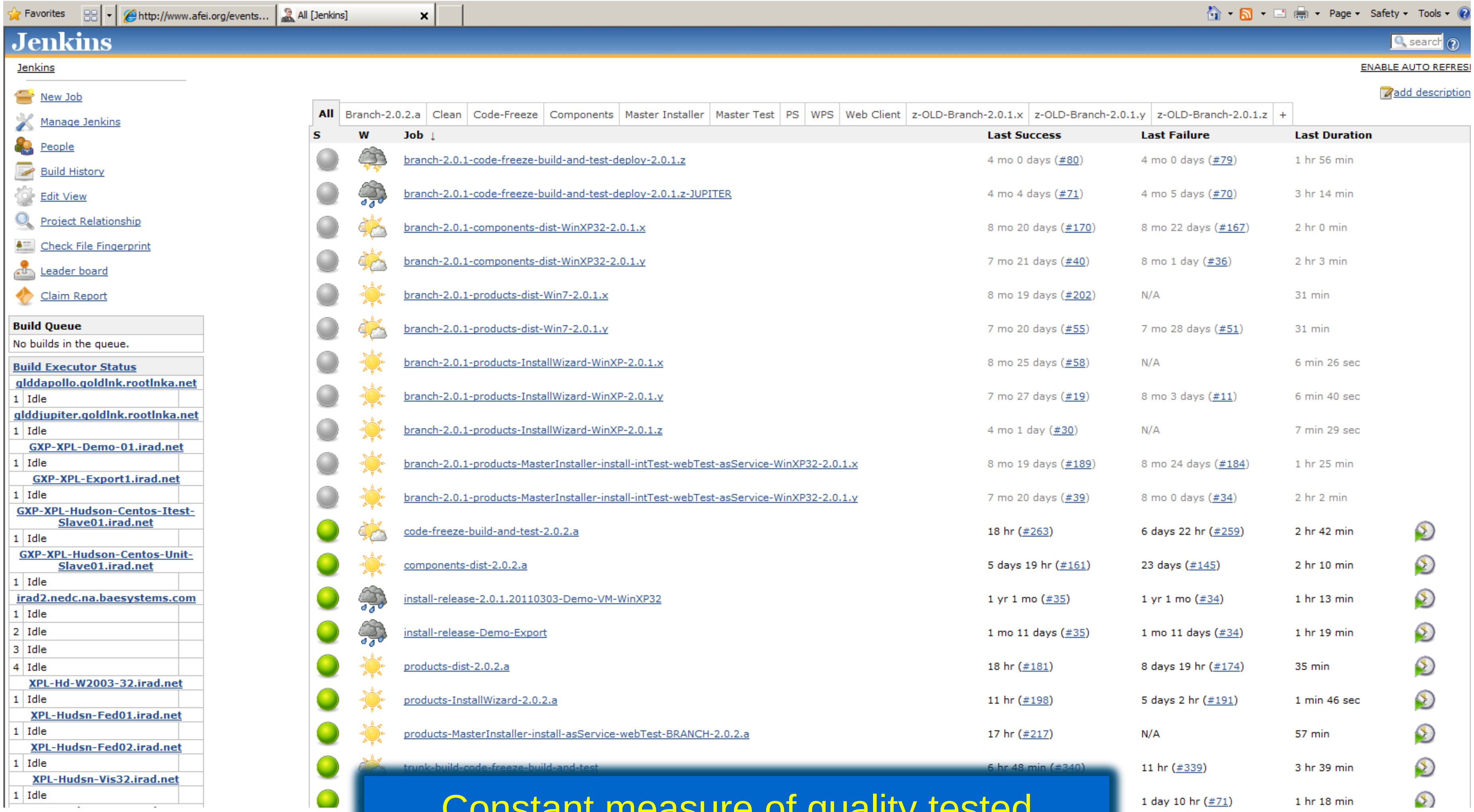
Continuous integration and test




5
If Build or SW Test Failed, GET IT FIXED NOW
Lead
Any Desktop
Build / Test Status Reports
1
Kickoff Nightly Build
4
Generate Daily Build/ Test Status Report
CI Server
3
Generate Check-in Status Report
3
Run SW Integration Tests
4
If Build or UT Failed, FIX IT NOW
SW Int. Test Status
Check-in Build / Unit Test Status
Build Status
Build / Test Instructions
Test System
Developer
Development System / IDE
Check_in
Deployment
2
Deploy Nightly Build
1
Check-in New Code
2
Check-In Triggers Quick Build and Unit Test
CM/Build Server

BAE SYSTEMS

Continuous Integration and Testing environment using COTS, Open Source CI tool



Jenkins

ENABLE AUTO REFRESI

- New Job
- Manage Jenkins
- People
- Build History
- Edit View
- Project Relationship
- Check File Fingerprint
- Leader board
- Claim Report

Build Queue

No builds in the queue.

Build Executor Status

glddapollo.goldlnk.rootlnka.net	
1	Idle
glddjupiter.goldlnk.rootlnka.net	
1	Idle
GXP-XPL-Demo-01.irad.net	
1	Idle
GXP-XPL-Export1.irad.net	
1	Idle
GXP-XPL-Hudson-Centos-Itest-Slave01.irad.net	
1	Idle
GXP-XPL-Hudson-Centos-Unit-Slave01.irad.net	
1	Idle
irad2.nedc.na.baesystems.com	
1	Idle
2	Idle
3	Idle
4	Idle
XPL-Hd-W2003-32.irad.net	
1	Idle
XPL-Hudsn-Fed01.irad.net	
1	Idle
XPL-Hudsn-Fed02.irad.net	
1	Idle
XPL-Hudsn-Vis32.irad.net	
1	Idle

add description

All | Branch-2.0.2.a | Clean | Code-Freeze | Components | Master Installer | Master Test | PS | WPS | Web Client | z-OLD-Branch-2.0.1.x | z-OLD-Branch-2.0.1.y | z-OLD-Branch-2.0.1.z | +

S	W	Job ↓	Last Success	Last Failure	Last Duration	
○	⛈	branch-2.0.1-code-freeze-build-and-test-deploy-2.0.1.z	4 mo 0 days (#80)	4 mo 0 days (#79)	1 hr 56 min	
○	🌧	branch-2.0.1-code-freeze-build-and-test-deploy-2.0.1.z-JUPITER	4 mo 4 days (#71)	4 mo 5 days (#70)	3 hr 14 min	
○	⛅	branch-2.0.1-components-dist-WinXP32-2.0.1.x	8 mo 20 days (#170)	8 mo 22 days (#167)	2 hr 0 min	
○	⛅	branch-2.0.1-components-dist-WinXP32-2.0.1.y	7 mo 21 days (#40)	8 mo 1 day (#36)	2 hr 3 min	
○	☀	branch-2.0.1-products-dist-Win7-2.0.1.x	8 mo 19 days (#202)	N/A	31 min	
○	⛅	branch-2.0.1-products-dist-Win7-2.0.1.y	7 mo 20 days (#55)	7 mo 28 days (#51)	31 min	
○	☀	branch-2.0.1-products-InstallWizard-WinXP-2.0.1.x	8 mo 25 days (#58)	N/A	6 min 26 sec	
○	☀	branch-2.0.1-products-InstallWizard-WinXP-2.0.1.y	7 mo 27 days (#19)	8 mo 3 days (#11)	6 min 40 sec	
○	☀	branch-2.0.1-products-InstallWizard-WinXP-2.0.1.z	4 mo 1 day (#30)	N/A	7 min 29 sec	
○	☀	branch-2.0.1-products-MasterInstaller-install-intTest-webTest-asService-WinXP32-2.0.1.x	8 mo 19 days (#189)	8 mo 24 days (#184)	1 hr 25 min	
○	☀	branch-2.0.1-products-MasterInstaller-install-intTest-webTest-asService-WinXP32-2.0.1.y	7 mo 20 days (#39)	8 mo 0 days (#34)	2 hr 2 min	
●	⛅	code-freeze-build-and-test-2.0.2.a	18 hr (#263)	6 days 22 hr (#259)	2 hr 42 min	▶
●	☀	components-dist-2.0.2.a	5 days 19 hr (#161)	23 days (#145)	2 hr 10 min	▶
●	🌧	install-release-2.0.1.20110303-Demo-VM-WinXP32	1 yr 1 mo (#35)	1 yr 1 mo (#34)	1 hr 13 min	▶
●	🌧	install-release-Demo-Export	1 mo 11 days (#35)	1 mo 11 days (#34)	1 hr 19 min	▶
●	☀	products-dist-2.0.2.a	18 hr (#181)	8 days 19 hr (#174)	35 min	▶
●	☀	products-InstallWizard-2.0.2.a	11 hr (#198)	5 days 2 hr (#191)	1 min 46 sec	▶
●	☀	products-MasterInstaller-install-asService-webTest-BRANCH-2.0.2.a	17 hr (#217)	N/A	57 min	▶
●	⛅	trunk-build-code-freeze-build-and-test	6 hr 48 min (#340)	11 hr (#339)	3 hr 39 min	▶
●	[illegible]	[illegible]	[illegible]	1 day 10 hr (#71)	1 hr 18 min	▶

Constant measure of quality tested

Agile with Continuous Integration and Test

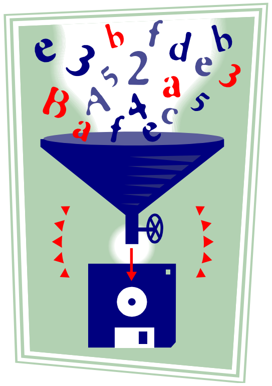
- When an capability development phase is planed, time is held “fixed” within the iterations (i.e., multiple four week Agile Scrums)
  - It becomes *obvious* at the daily meetings (both by CI&T automated results and discussions with the programmers) where the difficulties are occurring
  - Peer / Team relationships foster a culture of feedback and improvement
  - Schedule adherence or deviations is near real-time

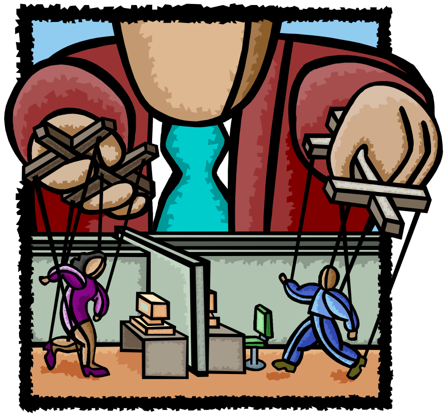
- Actual cost are accounted for daily and reviewed weekly
  - Actual costs rise and fall proportionally to the degree of difficulty difference from the “Planned” tasks

- Quality of the software under development is monitored nightly by CI&T
  - Metrics collected show trends and areas of concern
  - Decisions can be made with insight about high risk areas

Earned Value Management conundrum with Agile

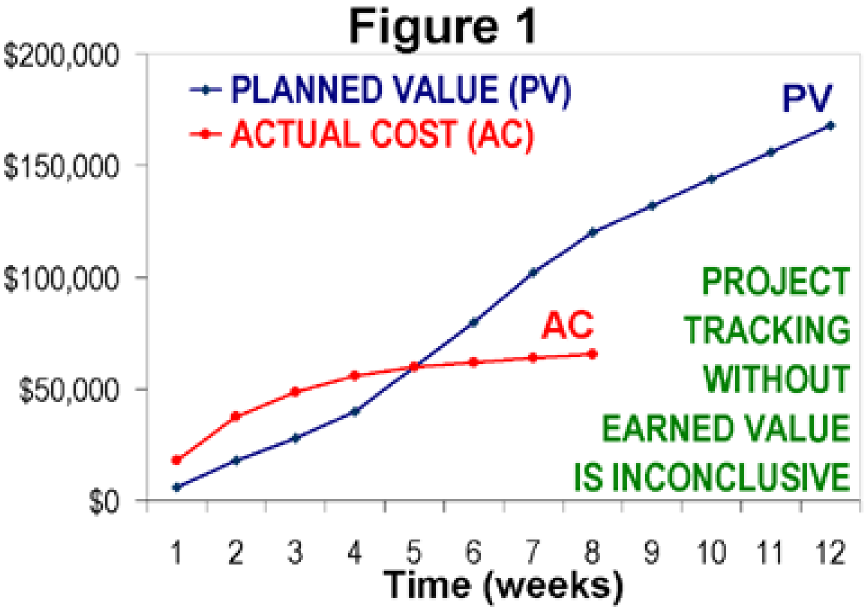
- Because EVM requires quantification of a project plan, it is often perceived to be inapplicable for Agile software development projects

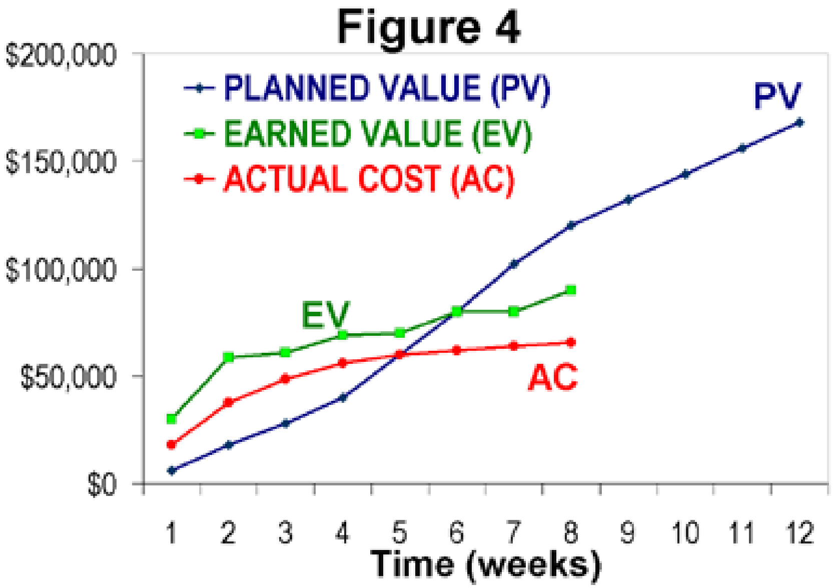




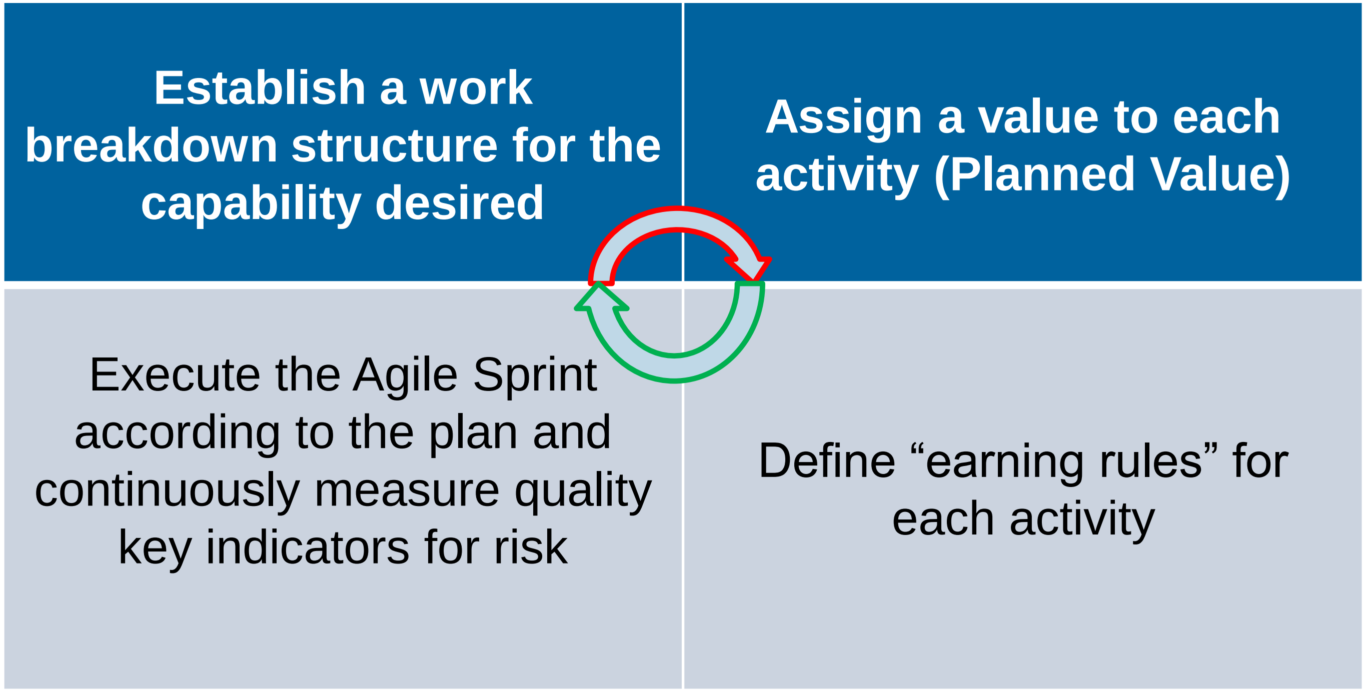
- However, another school of thought holds that all work can be planned, even if in weekly time boxes or other short increments

With EV the status is inconclusive





Agile EV



Establish a work breakdown structure for the capability desired

Assign a value to each activity (Planned Value)

Execute the Agile Sprint according to the plan and continuously measure quality key indicators for risk

Define “earning rules” for each activity

True understanding of cost and schedule performance *relies first on measuring quality objectively*

Risk becomes visible

$$PVr = PVi_1 + PVi_2 + \ldots . PVi_n$$

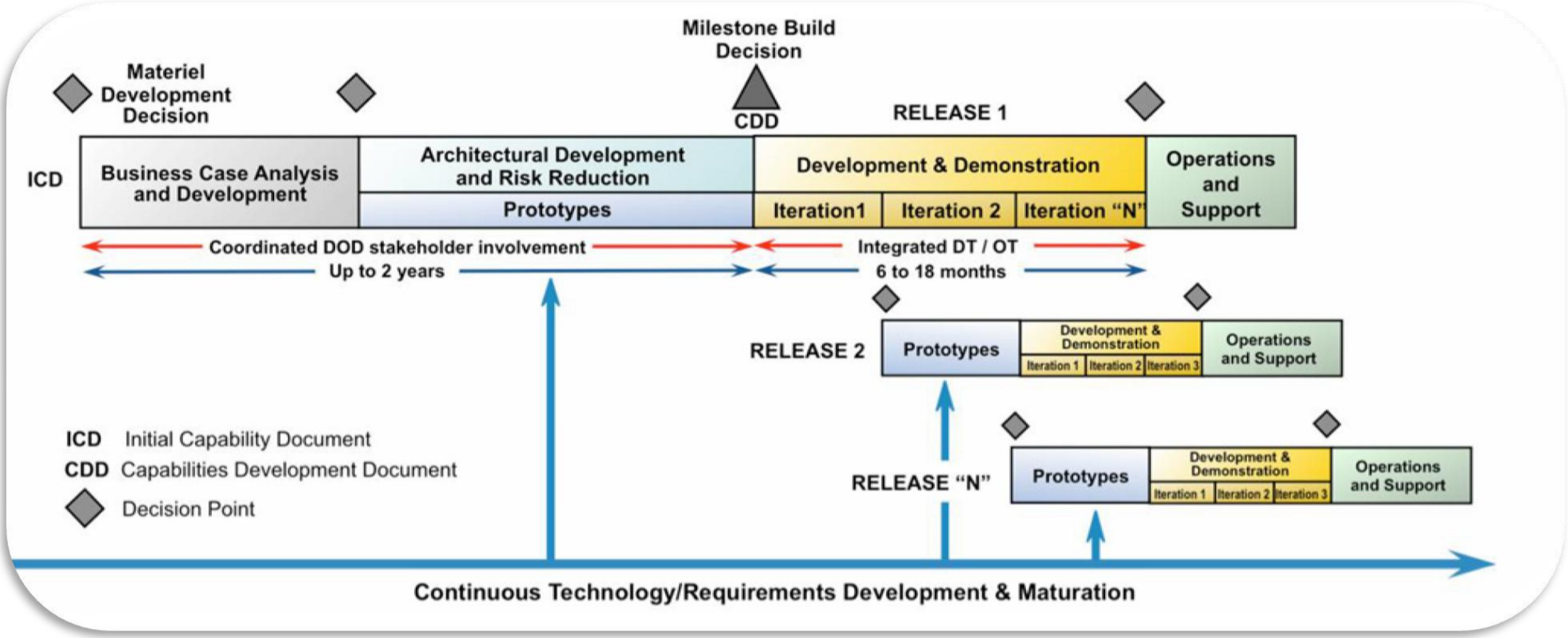
Planned Value of the release (PVr) is maintained constant (constrained cost and time) by allowing PVi to be modified based on efficiencies, enlightenment, or risks realized/avoided (both pos / neg)

Since agile EV is the measure of schedule and cost adherence to the PVi, software intensive risks becomes visible based upon the *degree of difficulty difference between the PVi and EVa*

$$EVa = \sum_{start}^{current} PVi$$

Change is coming

FY2010 NDAA Section 804



ICD Initial Capability Document

CDD Capabilities Development Document

◆ Decision Point

The Next Sprint has already started

☑ constant measure of schedule achieved

☑ constant measure of actual cost incurred

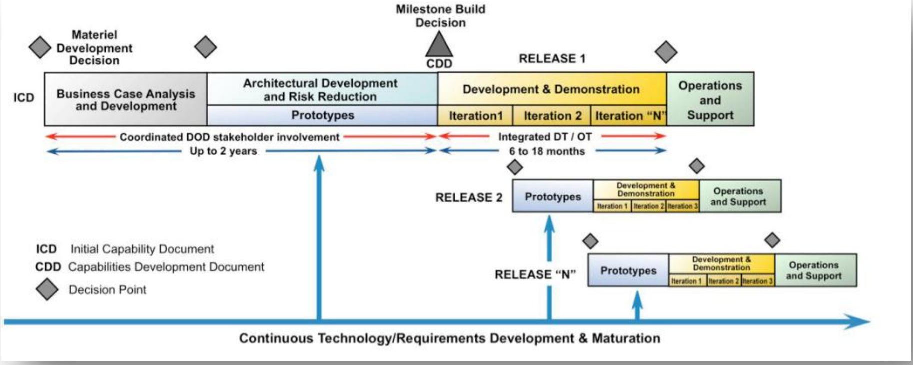
☑ constant measure of quality tested

☑ Agile Earned Value

Risk is actionable with Agile processes

Innovation comes with agile EVMS

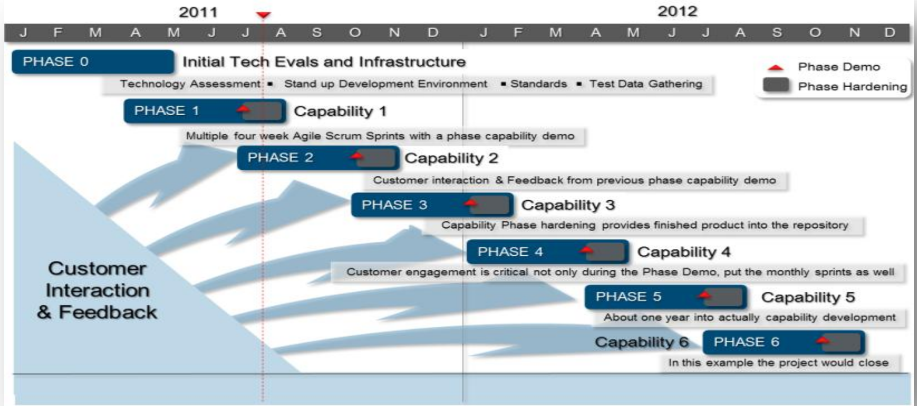
A project plan is developed that identifies work to be accomplished (work breakdown structure) both at the release level and with specific details at the iteration levels


Milestone Build Decision
Materiel Development Decision
CDD
RELEASE 1
ICD
Business Case Analysis and Development
Architectural Development and Risk Reduction
Prototypes
Development & Demonstration
Iteration1
Iteration 2
Iteration "N"
Operations and Support
Coordinated DOD stakeholder involvement
Up to 2 years
Integrated DT / OT
6 to 18 months
RELEASE 2
Prototypes
Development & Demonstration
Iteration 1
Iteration 2
Iteration 3
Operations and Support
RELEASE "N"
Prototypes
Development & Demonstration
Iteration 1
Iteration 2
Iteration 3
Operations and Support
ICD Initial Capability Document
CDD Capabilities Development Document
Decision Point
Continuous Technology/Requirements Development & Maturation

A valuation of planned work at the release level, Planned Value "release" (PVr)

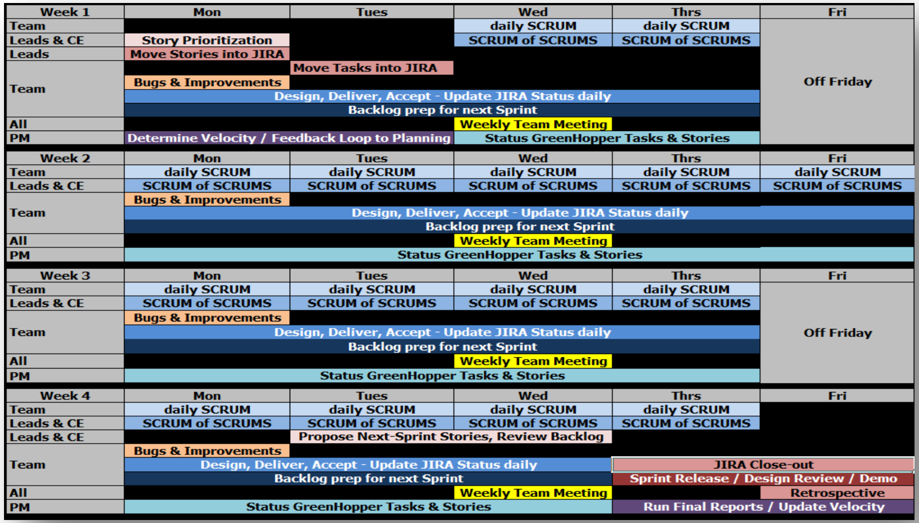
A pre-defined set of "earning rules" (metrics) to quantify the accomplishment of work called Earned Value (EV)

Prior to the start of the next capability iteration phase, the PVr remains constant (can always tell where you have been), but the Planned Value iteration(PVi) is adjustable to ensure that EVa is measuring the right items



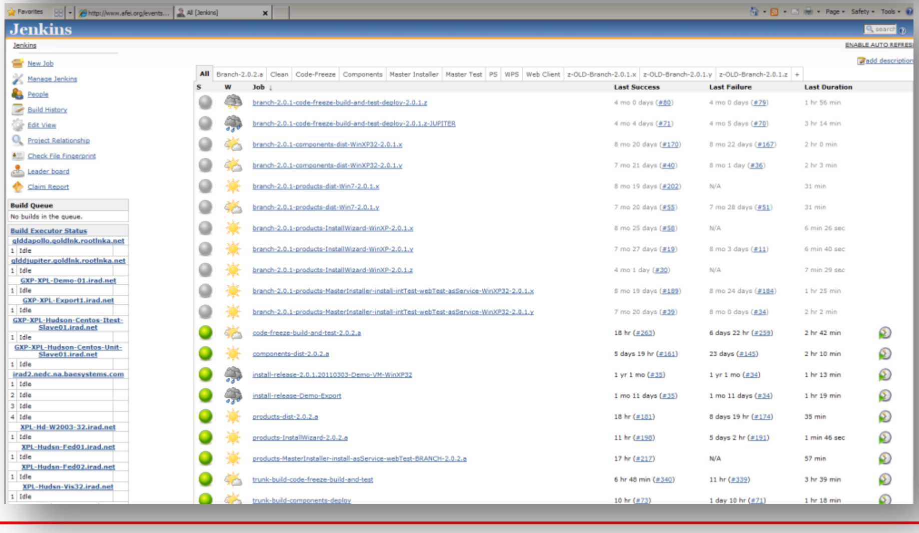
Innovation comes with agile EVMS

Agile Scrums within an iteration phase provides actual cost information (c) and schedule (s) earnings as they are tightly coupled to the quality test event



Week 1	Mon	Tues	Wed	Thrs	Fri
Team			daily SCRUM	daily SCRUM	Off Friday
Leads & CE	Story Prioritization		SCRUM of SCRUMS	SCRUM of SCRUMS	
Leads	Move Stories into JIRA				
Team		Move Tasks into JIRA			
	Bugs & Improvements				
	Design, Deliver, Accept - Update JIRA Status daily				
	Backlog prep for next Sprint				
All			Weekly Team Meeting		
PM	Determine Velocity / Feedback Loop to Planning		Status GreenHopper Tasks & Stories		
Week 2	**Mon**	**Tues**	**Wed**	**Thrs**	**Fri**
Team	daily SCRUM	daily SCRUM	daily SCRUM	daily SCRUM	daily SCRUM
Leads & CE	SCRUM of SCRUMS	SCRUM of SCRUMS	SCRUM of SCRUMS	SCRUM of SCRUMS	SCRUM of SCRUMS
Team	Bugs & Improvements				
	Design, Deliver, Accept - Update JIRA Status daily				
	Backlog prep for next Sprint				
All			Weekly Team Meeting		
PM	Status GreenHopper Tasks & Stories				
Week 3	**Mon**	**Tues**	**Wed**	**Thrs**	**Fri**
Team	daily SCRUM	daily SCRUM	daily SCRUM	daily SCRUM	Off Friday
Leads & CE	SCRUM of SCRUMS	SCRUM of SCRUMS	SCRUM of SCRUMS	SCRUM of SCRUMS	
Team	Bugs & Improvements				
	Design, Deliver, Accept - Update JIRA Status daily				
	Backlog prep for next Sprint				
All			Weekly Team Meeting		
PM	Status GreenHopper Tasks & Stories				
Week 4	**Mon**	**Tues**	**Wed**	**Thrs**	**Fri**
Team	daily SCRUM	daily SCRUM	daily SCRUM	daily SCRUM	
Leads & CE	SCRUM of SCRUMS	SCRUM of SCRUMS	SCRUM of SCRUMS	SCRUM of SCRUMS	
Leads & CE		Propose Next-Sprint Stories, Review Backlog			
Team	Bugs & Improvements				
	Design, Deliver, Accept - Update JIRA Status daily			JIRA Close-out	
	Backlog prep for next Sprint			Sprint Release / Design Review / Demo	
All			Weekly Team Meeting		Retrospective
PM	Status GreenHopper Tasks & Stories			Run Final Reports / Update Velocity	

Continuous integration and test using automated test tools provides "continuous" monitoring of the build progress and flags areas of concern prior to the Scrum demo event (Q)



Build Queue

No builds in the queue.

Build Executor Status

glddapollo.goldlnk.rootlnka.net	
1	Idle
glddjupiter.goldlnk.rootlnka.net	
1	Idle
GXP-XPL-Demo-01.irad.net	
1	Idle
GXP-XPL-Export1.irad.net	
1	Idle
GXP-XPL-Hudson-Centos-Itest-Slave01.irad.net	
1	Idle
GXP-XPL-Hudson-Centos-Unit-Slave01.irad.net	
1	Idle
irad2.nedc.na.baesystems.com	
1	Idle
2	Idle
3	Idle
4	Idle
XPL-Hd-W2003-32.irad.net	
1	Idle
XPL-Hudsn-Fed01.irad.net	
1	Idle
XPL-Hudsn-Fed02.irad.net	
1	Idle
XPL-Hudsn-Vis32.irad.net	
1	Idle

S	W	Job ↓	Last Success	Last Failure	Last Duration
●		branch-2.0.1-code-freeze-build-and-test-deploy-2.0.1.z	4 mo 0 days (#80)	4 mo 0 days (#79)	1 hr 56 min
●		branch-2.0.1-code-freeze-build-and-test-deploy-2.0.1.z-JUPITER	4 mo 4 days (#71)	4 mo 5 days (#70)	3 hr 14 min
●		branch-2.0.1-components-dist-WinXP32-2.0.1.x	8 mo 20 days (#170)	8 mo 22 days (#167)	2 hr 0 min
●		branch-2.0.1-components-dist-WinXP32-2.0.1.y	7 mo 21 days (#40)	8 mo 1 day (#36)	2 hr 3 min
●		branch-2.0.1-products-dist-Win7-2.0.1.x	8 mo 19 days (#202)	N/A	31 min
●		branch-2.0.1-products-dist-Win7-2.0.1.y	7 mo 20 days (#55)	7 mo 28 days (#51)	31 min
●		branch-2.0.1-products-InstallWizard-WinXP-2.0.1.x	8 mo 25 days (#58)	N/A	6 min 26 sec
●		branch-2.0.1-products-InstallWizard-WinXP-2.0.1.y	7 mo 27 days (#19)	8 mo 3 days (#11)	6 min 40 sec
●		branch-2.0.1-products-InstallWizard-WinXP-2.0.1.z	4 mo 1 day (#30)	N/A	7 min 29 sec
●		branch-2.0.1-products-MasterInstaller-install-intTest-webTest-asService-WinXP32-2.0.1.x	8 mo 19 days (#189)	8 mo 24 days (#184)	1 hr 25 min
●		branch-2.0.1-products-MasterInstaller-install-intTest-webTest-asService-WinXP32-2.0.1.y	7 mo 20 days (#39)	8 mo 0 days (#34)	2 hr 2 min
●		code-freeze-build-and-test-2.0.2.a	18 hr (#263)	6 days 22 hr (#259)	2 hr 42 min
●		components-dist-2.0.2.a	5 days 19 hr (#161)	23 days (#145)	2 hr 10 min
●		install-release-2.0.1.20110303-Demo-VM-WinXP32	1 yr 1 mo (#35)	1 yr 1 mo (#34)	1 hr 13 min
●		install-release-Demo-Export	1 mo 11 days (#35)	1 mo 11 days (#34)	1 hr 19 min
●		products-dist-2.0.2.a	18 hr (#181)	8 days 19 hr (#174)	35 min
●		products-InstallWizard-2.0.2.a	11 hr (#198)	5 days 2 hr (#191)	1 min 46 sec
●		products-MasterInstaller-install-asService-webTest-BRANCH-2.0.2.a	17 hr (#217)	N/A	57 min
●		trunk-build-code-freeze-build-and-test	6 hr 48 min (#340)	11 hr (#339)	3 hr 39 min
●		trunk-build-components-deploy	10 hr (#73)	1 day 10 hr (#71)	1 hr 18 min